1. 上课须知
2. 上次内容总结
3. 上次作业
4. 本次内容预告
    4. 1. Hive的架构设计和SQL语句复习总结
        4. 1. 1. Hive 离线数仓工具
        4. 1. 2. 数仓的概念
        4. 1. 3. 架构和原理
        4. 1. 4. 四个概念
        4. 1. 5. 支持的SQL语法总结
    4. 2. Hive的环境部署 和 语法详解
    4. 3. Hive的JSON，Transform语法，复杂数据类型处理解读
        4. 3. 1. 解析JSON支持
        4. 3. 2. Transform语法
        4. 3. 3. 复杂数据类型支持
    4. 4. Hive的窗口分析函数详解
    4. 5. Hive的三大典型需求分析和企业案例完整实现
        4. 5. 1. 自连接
        4. 5. 2. TopN
        4. 5. 3. 行列转换
    4. 6. Hive的典型常见面试题型分析和思路总结
    4. 7. Hive全局排序
5. 总结
6. 作业
7. 彩蛋

1. 上课须知

课程主题：企业最佳实战案例和面试题型解析
上课时间：20:00 - 23:00
课件休息：21:30 左右 休息10分钟
课前签到：如果能听见音乐，能看到画面，请在直播间扣 666 签到！

2. 上次内容总结

MapReduce的源码分析：

```
1、MapReduce的程序执行流程分析
2、MapReduce的组件分析
3、MapReduce的编程模型分析
4、MapReduce的任务提交分析
5、MapReduce的逻辑切片分析
6、MapReduce的MapTask执行分析
7、MapReduce的Shuffle分析
8、MapReduce的ReduceTask分析
```

MapReduce 是第一代（基于磁盘的分布式计算编程模型（v2）+ 计算引擎（v1））

MapReduce 仅仅只是提供了一套用于编写分布式计算应用程序的 API

问题：为什么有reducer，mapreduce框架，就一定会按照 key 进行排序呢?

3. 上次作业

使用 MapReduce 编程模型，实现海量数据的分布式排序

```
三个要求：
1、海量数据，MapReduce编程模型
2、提交到集群运行
3、不能使用一个ReduceTask运行
```

4. 本次内容预告

```
1、Hive的架构设计和SQL语句复习总结
2、Hive的窗口分析函数详解
3、Hive的JSON，Transform语法，复杂数据类型处理解读
4、Hive的三大典型需求分析和企业案例完整实现
5、Hive的典型常见面试题型分析和思路总结
6、Hive全局排序
```

请注意：最后有彩蛋!

4.1. Hive的架构设计和SQL语句复习总结

4.1.1. Hive 离线数仓工具

概念：http://hive.apache.org/

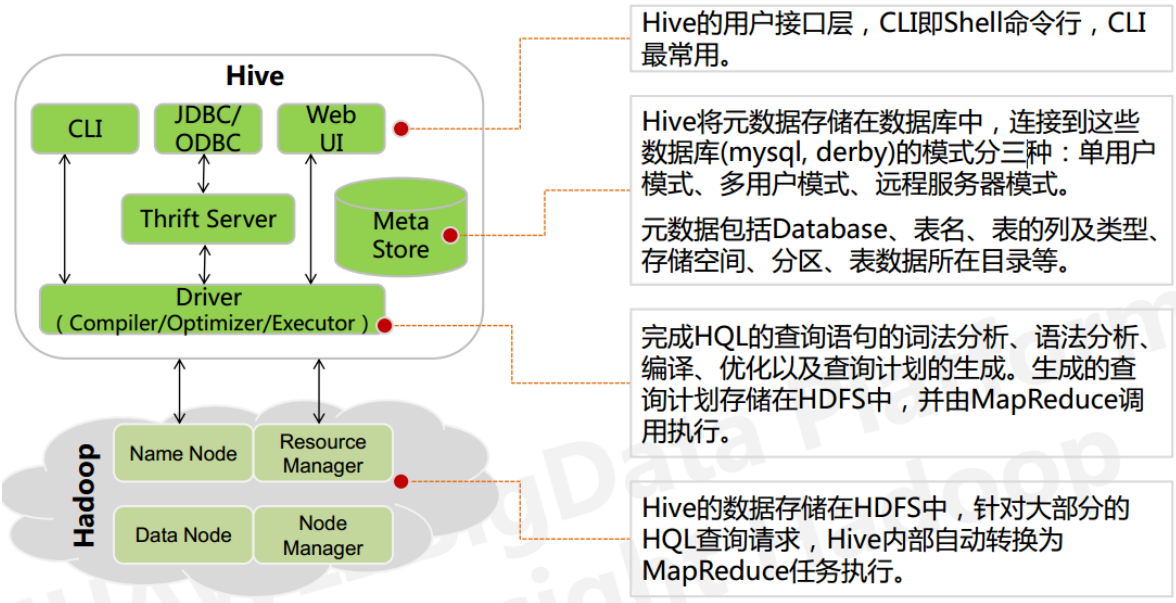
1、Hive由Facebook实现并开源
2、Hive是基于Hadoop的一个数据仓库工具
3、Hive存储的数据其实底层存储在HDFS上
4、Hive将HDFS上的结构化的数据映射为一张数据库表，类似于excel或者msyql的表
5、Hive提供HQL(Hive SQL)查询功能
6、Hive的本质是将SQL语句转换为MapReduce任务运行，使不熟悉MapReduce的用户很方便地利用HQL处理和计算HDFS上的结构化的数据，适用于离线的批量数据计算
7、Hive使用用户可以极大简化分布式计算程序的编写，而将精力集中于业务逻辑

总结：Hive依赖于HDFS存储数据，Hive将HQL转换成MapReduce执行，所以说Hive是基于Hadoop的一个数据仓库工具，实质就是一款基于HDFS的MapReduce计算框架，对存储在HDFS中的数据进行分析和管理。

4.1.2. 数仓的概念

数据仓库之父比尔•恩门（Bill Inmon）在1991年出版的"Building the Data Warehouse"（《建立数据仓库》）一书中所提出的定义被广泛接受：
数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策(Decision Making Support)。

4.1.3. 架构和原理



Hive内部四大组件：**Driver驱动器：编译器Compiler，优化器Optimizer，执行器Executor**

Driver组件完成HQL查询语句从词法分析，语法分析，编译，优化，以及生成逻辑执行计划的生成。生成的逻辑执行计划存储在HDFS中，并随后由MapReduce调用执行。

4.1.4. 四个概念

```
1、数据库
2、数据仓库
3、数据湖
4、数据中台
```

4.1.5. 支持的SQL语法总结

支持的语法：

```
1、select * from db.table1
2、select count(distinct uid) from db.table1
3、支持select、union all、join(left、right、full join)、like、where、having、limit等标准语法
4、支持各种普通函数，聚合函数、表格函数
5、支持json解析
6、支持函数自定义：UDF（User Defined Function）/ UDAF/UDTF
7、不支持update和delete，不管新老版本都不支持，而且没有支持的必要！
8、hive虽然支持in/exists（老版本不支持），但是hive推荐使用semi join的方式来代替实现，而且效率更高。
9、支持case … when …
```

不支持的语法：

```
1、支持 and多条件join，不支持 or 多条件join
select a.*, b.* from a join b on a.id = b.id and a.name = b.name;  √√√√√
select a.*, b.* from a join b on a.id = b.id or a.name = b.name;   xxxxx

2、默认情况下，不支持笛卡尔积
select a.*, b.* from a, b;
```

你的使用原则：绝大部分的SQL，常用的SQL语法，都是支持的！

4.2. Hive的环境部署 和 语法详解

提供了资料，在此不细讲

4.3. Hive的JSON，Transform语法，复杂数据类型处理解读

4.3.1. 解析JSON支持

现有原始 JSON 数据（rating.json）如下：

```
{"movie":"1193","rate":"5","timeStamp":"978300760","uid":"1"}
{"movie":"661","rate":"3","timeStamp":"978302109","uid":"1"}
{"movie":"914","rate":"3","timeStamp":"978301968","uid":"1"}
{"movie":"3408","rate":"4","timeStamp":"978300275","uid":"1"}
{"movie":"2355","rate":"5","timeStamp":"978824291","uid":"1"}
.....
{"movie":"1197","rate":"3","timeStamp":"978302268","uid":"1"}
{"movie":"1287","rate":"5","timeStamp":"978302039","uid":"1"}
{"movie":"2804","rate":"5","timeStamp":"978300719","uid":"1"}
{"movie":"594","rate":"4","timeStamp":"978302268","uid":"1"}
```

现在需要将数据导入到hive仓库中，并且最终要得到这么一个结果：

movie	rate	timeStamp	uid
1193	5	978300760	1

分析解决方案：

```
1、尝试使用内置函数搞定。get_json_object()
2、如果你不曾知道有内置函数支持，尝试编写UDF
3、尝试编写处理脚本（shell，python）来处理json，依然还是在hive中。
```

解决：

1、先加载 rating.json 文件到 hive 的一个原始表 rate_json_1_raw

```
create database if not exists jsondb;
use jsondb;
drop table if exists rate_json_1_raw;
create table rate_json_1_raw(line string) row format delimited;
load data local inpath '/home/bigdata/rating.json' into table rate_json_1_raw;
select line from rate_json_1_raw limit 3;
```

2、创建 rate_json_2_detail 这张表用来存储解析 json 出来的字段：

```
create table rate_json_2_detail (movie int, rate int, unixtime int, userid int)
row format delimited fields terminated by '\t';
```

解析 json，得到结果之后存入 rate_json_2_detail表：

```
insert into table rate_json_2_detail
select
get_json_object(line,'$.movie') as moive,
get_json_object(line,'$.rate') as rate,
get_json_object(line,'$.timeStamp') as unixtime,
get_json_object(line,'$.uid') as userid
from rate_json_1_raw;
```

3、检查

```
select * from rate_json_2_detail limit 3;
```

结果：

```
moive   rate    unixtime        userid
1193    5       978300760       1
661     3       978302109       1
914     3       978301968       1
```

4.3.2. Transform语法

需求：把时间戳变成星期编号（0-7）

解决方案：

```
1、内置函数
2、自定义函数
3、使用Transform语法：  使用外部脚本充当自定义函数   y=f(x)
```

方案：

```
1、函数 或者 自定义函数 轻松实现转换
2、运维人员编写 Shell 脚本，或者 Python 脚本，hive 来调用执行转换。这就是 transform
```

先编辑一个 python 脚本文件：weekday_mapper.py

```
#!/bin/python

import sys
import datetime
for line in sys.stdin:
  line = line.strip()
  movie,rate,unixtime,userid = line.split('\t')
  weekday = datetime.datetime.fromtimestamp(float(unixtime)).isoweekday()
  print '\t'.join([movie, rate, str(weekday), userid])
```

然后，将文件加入 hive 的 classpath：

```
hive> add file /home/bigdata/weekday_mapper.py;
```

创建最后的用来存储调用 python 脚本解析出来的数据的表：rate_json_3_result

```
hive> create table rate_json_3_result(movie int, rate int, weekday int, userid
int) row format delimited fields terminated by '\t';
```

执行转换：

```
hive> insert into table rate_json_3_result
select transform(movie,rate,unixtime,userid)
using 'python weekday_mapper.py'
as(movie,rate,weekday,userid)
from rate_json_2_detail;
```

最后查询看数据是否正确：

```
hive> select * from rate_json_3_result limit 10;
hive> select distinct(weekday) from rate_json_3_result;
```

4.3.3. 复杂数据类型支持

复杂数据类型包括数组（ARRAY）、映射（MAP）和结构体（STRUCT），具体如下所示：

类型	使用方式	访问方式	数据示例
array	favors array<string>	column[1]	['apple','orange','mango']
map	scores map<string, float>	column['username']	"math":87,"english":77
struct	address struct<bb: string, cc: float>	column.name	"北京","海淀",135..., 211023

例子：

```
CREATE TABLE student_complext_type(
    name STRING,
    favors ARRAY<STRING>,
    scores MAP<STRING, FLOAT>,
    address STRUCT<province:STRING, city:STRING, phone:STRING, zip:INT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ';'
MAP KEYS TERMINATED BY ':' ;
```

4.4. Hive的窗口分析函数详解

见文档：Hive窗口分析函数全解

```
// 窗口内的数据做什么逻辑操作， avg max, min, 。。。
SUM(pv)
// 定义窗口分组和排序的逻辑
OVER(PARTITION BY cookieid ORDER BY createtime
// 定义窗口的大小： (3 PRECEDING  +  current row + ?)
ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS pv4
// 窗口的定义：1 PRECEDING + CURRENT ROW + 4 following
ROWS BETWEEN 1 PRECEDING and 4 following as pv44

A：有三种写法：
unbounded preceding          从当前组的第一条开始
3 preceding                  从当前行数 3 行
current row                  当前行

B：有三种写法：
unbounded following          到分组的最后一行结束
```

```
4 following                     当前行的后四行
current row                     当前行
```

第二类：需求：我要求全校的每个班的第一名

```
select max(score), class from school group by class group by class;
```

需求：我要求全校每个班的前3名！

包含两个逻辑：

1、必然要分组

2、组内数据必须要排序，而且排序之后，只取3条记录

```
class1:     100 99  98  98  98  97
class2:     100 98  98  98  97  96
.....
```

到底怎么取数据，还跟具体的应用场景有关系？

```
1、成绩         100 99  98  98  98
2、热搜         100 98  98
```

提供两种解决方案：

1、join查询

第一步：先求出每个班的最高的3个成绩

第二步：使用join查询求得，等于这些成绩的学生的信息

```
select .... from grade join maxscore on classid = classid and score =
maxscore;
```

2、窗口函数：给每一个窗口内的数据，排序之后，生成一个由低到高的一个顺序编号

rank

row_number

dense_rank

	classid	row_number	rank	dense_rank
100	1	1	1	1
99	1	2	2	2
98	1	3	3	3
98	1	4	3	3
98	1	5	3	3
97	1	6	6	4
100	2	1		
98	2	2		
98	2	3		
97	2	4		

求出每个班的最高的3个成绩： `select * from result where dense_rank <= 3;`

row_number 按顺序编号，不留空位

rank 按顺序编号，相同的值编相同号，留空位

dense_rank 按顺序编号，相同的值编相同的号，不留空位

新的需求：求没个班的 排名前1/3 的人

4.5. Hive的三大典型需求分析和企业案例完整实现

4.5.1. 自连接

题目见：Hive经典企业案例--自连接.pdf

两种解决方案：

```
1、自连接
2、窗口函数
```

```
create database if not exists nx_hivedb;
use nx_hivedb;
drop table if exists nx_example1_raw;
create table nx_example1_raw(name string, month string, pv int) row format
delimited fields terminated by ",";
load data local inpath "/home/bigdata/nx_example1_raw.txt" into table
nx_example1_raw;
select * from nx_example1_raw;

create table nx_example1_ods as select name, month,  sum(pv) as tpv from
nx_example1_raw group by name, month;

// 自连接实现：
create table nx_example1_3 as select
a.name as aname, a.month as amonth, a.tpv as apv,
b.name as bname, b.month as bmonth, b.tpv as bpv
from nx_example1_ods a join nx_example1_ods b
on a.name = b.name;

select * from nx_example1_3 ;

create table  nx_example1_4 as select bname, bmonth, bpv, sum(apv) as sumpv,
max(apv) as maxpv from nx_example1_3 where amonth <= bmonth group by bname,
bmonth, bpv;

select * from nx_example1_4;

// 窗口实现：
create table nx_example1_5 as
select name, month, tpv,
sum(tpv) over (partition by name order by month rows between unbounded preceding
and current row) as maxpv,
max(tpv) over (partition by name order by month rows between unbounded preceding
and current row) as sumpv
from nx_example1_ods;
```

4.5.2. TopN

题目见：Hive经典企业案例--TopN案例.pdf

```
create table exercise_topn_dw as
select a.id as id, a.name as name, a.age as age,
favor_view.f as favor
from exercise_topn a
LATERAL view explode(split(a.favors, "-")) favor_view as f;

select a.id as id, a.name as name, a.age as age, a.favor as favor
from (
    select
    b.id as id, b.name as name, b.age as age, b.favor as favor,
    row_number() over (partition by b.favor order by b.age desc) as rank
    from exercise_topn_dw b
) a where a.rank <= 2;
```

4.5.3. 行列转换

题目见：Hive经典企业案例--行列转换.pdf

```
create table exercise_course_result2 as
select
sid,
max(case course when "yuwen" then score else 0 end) as yuwen,
max(case course when "shuxue" then score else 0 end)  as shuxue,
max(case course when "yingyu" then score else 0 end)  as yingyu
from exercise_course group by sid;

select sid from exercise_course_result2 where yuwen < shuxue;
```

4.6. Hive的典型常见面试题型分析和思路总结

1、注意：HQL语法分支顺序：编辑顺序：select ... from ... join ... on ... where .... group by .... having .... order by ..... limit ....

2、注意：链接查询（inner join，outer join , semi join）semi join 这个是 in/exists 的高效实现

3、注意：union 和 case .... when .... 很少用，但是有奇效！

4、注意：分组聚合 (count, distinct, sum, max, min, avg) + group by 和 explode （炸裂，拆分，表格生成）

5、注意：窗口函数（五大类，sum() over (), row_number() over ())

6、注意：需求关键字：每，各，分别，等使用 group by，并且注意 select ... group by ... order by .. 三种的执行顺序 和列别名的使用 (执行顺序：group by ---> select ---> order by) select 能起别名

```
1、因为 select 能起别名，group by 是不能使用别名的
2、order by 在 select 之后，所以 select 已经选择了对应的一些资料，所以order by的字段必须在 select 的范围里面。
```

7、注意：如果分组内，还有更细节更复杂（简单聚合不能满足要求）的逻辑处理，一般使用窗口函数。

```
如果逻辑上，是要分组的，但是简单的 group by 却又不能实现，则需要使用窗口分析！
```

8、注意：如果某些数据行，会在多个分组内被多次使用，注意使用 join链接 和 窗口函数

```
案例1 这种场景！
```

4.7. Hive全局排序

见：Hive经典企业案例--全局排序.pdf

```
MR程序实现海量数据的分布式排序！
写SQL
```

细节：如果使用 order by 来做，最终就是一个 reduceTask 来做，所以当数据量特别大的时候，肯定行不通。

方案细节：必然选择多个 reduceTask + sort by 做局部排序

```
set mapreduce.job.reduces = 10;
select id,name,sex,age,department from student sort by age desc;
```

并不能实现全局排序，稍稍改变就可以了！因为默认的数据分区是：Hash散列。

必要条件：只要能保证，第一个分区的所有数据，小于第二个分区，第二个分区的所有数据小于第三个分区

..... 最终的实现思路： 把 Hash散列 改成 范围分区！ + 分区降序排序，大的数据放第一个分区

0-100, 100-200, 200-300,.....

```
set mapreduce.job.reduces = 3;

insert overwrite directory "hdfs://hadoop277ha/hive_student_out_order3"
select
id,name,sex,age,department from exercise_student distribute by (
case
when age > 20 then 0
when age > 18 then 1
else 2
end)
sort by age desc;
```

有一个问题：你怎么就确定这三个分桶的界限是：20,18 呢?

有可能出现的问题：这三个桶中的数据分布不均匀！

```
1、先确定 max, min
2、然后通过抽样了解数据分布规律
    hive采样：
    1、全量数据取 5%
    2、全量数据取 100条
    3、全量数据取 100M
```

这两个操作，联合完成，范围分区

5. 总结

总结：清楚的掌握 Hive 的 HQL 语法和常见 编程案例 题型解析，掌握丰富的解题技巧。

```
1、Hive的架构设计和SQL语句复习总结
2、Hive的窗口分析函数详解
3、Hive的JSON，复杂数据类型处理解读
4、Hive的三大典型需求分析和企业案例完整实现
5、Hive的典型常见面试题型分析和思路总结
6、Hive全局排序
```

6. 作业

见文档：Hive最终大作业.pdf

7. 彩蛋

见文档：Hive经典企业案例--二进制相关.md